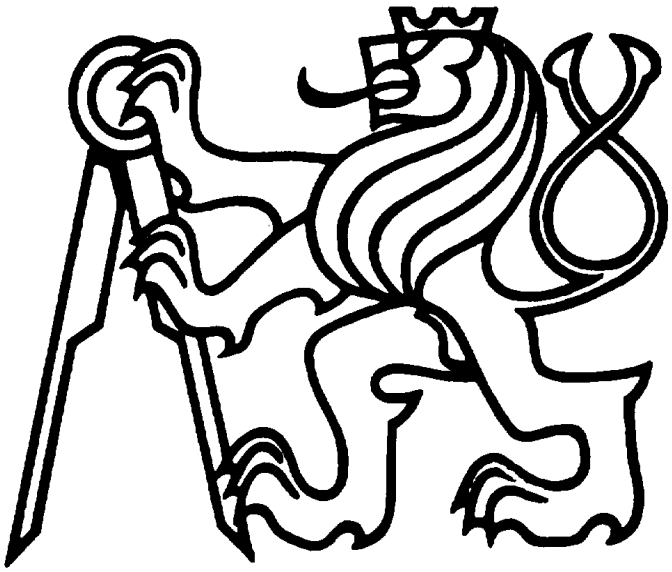


Master Thesis

Generalized Platform for Creating of Testing Games

Rafael Dorado

Index

Index of Illustrations

Index of Tables

1. Abstract

The thesis is an extension and a generalization of a previous theses done by Antonio Carro and Eduard Martínez. The goal of this work is to create a virtual platform where the teachers will be able to define different structures for the tests,create tests and follow the students progress.
This virtual platform will have an architecture of server-client allowing the platform independence and the future development of mobile clients also the possible integration of other e-learning platforms in the market. The main programming languages will be PHP[i], Javascript[ii], CSS[iii] for the Internet platform and web services.

2. Overview and Motivation

After three different attempts of developing this project, it is obviously difficult to find a possible innovation rather than just finishing what it is been started.
Nevertheless the splitting of this project in two different ones has provided some advantages but introduced some problems too.

One of the advantages of continuing a previous work, is that we can use all the good ideas that have been developed in the past years. Taking the best of them and solving the problems that have been appearing.
At first sight the most important factor of this Thesis is to join the previous work and create a complete system with a fully documented communication services between the different software components.
Due to the fact that they were two different theses programmed with different programming languages the existing implementation is lacking of a bit of consistency and documentation. By this we mean that is really complicated to continue with the current work, expand the software and in the future maintaining it.
Eduard's Thesis is focused on the Web Platform, it is developed in Java and a framework in order to help to create the database access and the **SOAP**[iv] web services we have decided to keep the structure of the database, and the templates of **exams**. But one of the first decisions we have made is to change Java[v] for PHP and Zend Framework[vi] because of our knowledge of the technologies and its simplicity in order to start developing in it without any previous experience.

The main objective of this project is to unify and generalize the previous work in order to make it available for many other different types of platforms, not just computers. It means that we have to create an environment whereas it will be integrated the software components that take care of the database of exams, users, questions and all the necessary data. We also have to create a series of web services in order to communicate between these parts and the clients.

For doing this we will discuss in this document the current technologies used, if they are the more suitable, which are the weakest points to improve and how we adapted the work done by now. Also we will document all the process and explain which points need to be studied and expanded and what problems we have been finding during our research and work.

3. State of Art

This project relies upon the previous work realized by Antonio Carro and Eduard Martínez. By this reason we are going to study which parts of their work we are going to maintain and how are they designed.

3.1. CastleQuest 2.0

(By Antonio Carro)

This current version of the game consists in a 3D world where the player can move around the map and interact with various types of objects. There are objects that will make questions appear, another that will help the student to finish the level.
Each level represents an exam, which is a group of tests created by some teacher, and each room represents one of these tests.

A test is defined like this:

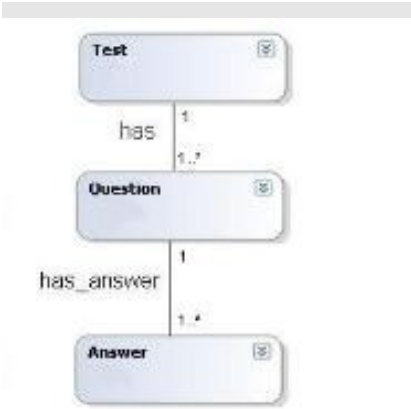


Illustration 3.1.1: Definition of Test (by Antonio Carro)

All tests have a group of questions and each question one or more correct answers.
It has also introduced the idea of test templates. These templates are useful to transform a test in a 3D world. They help to simplify the process of mapping a group of questions in an interesting and playable game map. It means that teachers have to choose one of these templates in order to create an exam.
Currently just one of these templates has been implemented and operative.

CastleQuest 2.0 should be able to connect through web services to the Testing Web Platform developed by Eduard Martínez.
The language used for developing this project was C#[vii] and the platform was Visual Studio. The engine responsible for rendering the 3D world is the Irrlicht.NET CP engine which is cross-platform and uses D3D[viii] and OpenGL[ix] for rendering.

In this thesis we focus on the creation of a computer-based client that will communicate with the server and allow the users play the game. We adopted the main ideas such as the way of defining tests, and how the template system of exams works.
We also utilized the idea of how to create the 2D visual map that we applied in the Map Editor that we will explain later.
In other words even if our goal was not to create a client, we found this thesis very useful to solve the logic of the games and also gave us a more clear view of what should look like a playable game.
By the way the dependency of an architecture (PC) and the use of not very well documented technologies made us to not consider this such a real client for our architecture, basically the work of adapting this client it is meaningless by the reasons we explained above.

3.2. CastleQuest Testing Web Application

(By Eduard Martínez)

This second project handle the way of how a teacher will create the test. This application also allows for the students to solve web-based tests and consult their previous results. It creates a XML model of the test that will be available by web services for the CastleQuest application. It also has a small statistics module in order to follow the progress of the students. There are 3 defined roles: teachers, students and an Administrator.
The next picture shows the diagram made by Eduard, explaining the different users of the application and how they interact. We can see the client CastleQuest developed by Antonio how it is connected.

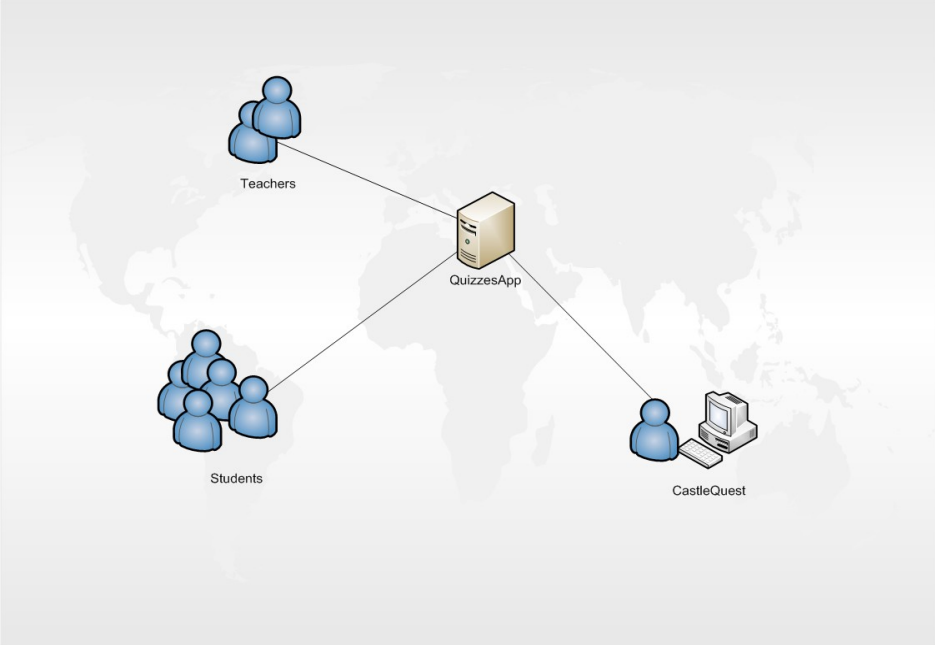


Illustration 3.2.1: User Interactions (by Eduard Martínez)

This piece of software is developed in Java and he decided to use a framework called Spring because it is functionalities and it is documentation. These are some of the main characteristics of Java and Spring[x] taken from the official website of Spring.

The communication system is implemented with XML[xi] and SOAP web services, It is a very common and well documented solution even though we decided to change them for REST[xii] web services protocol. We will discuss the differences between the two methods and why we changed the technologies.

3.2.1. Spring Features

Spring is a layered Java/J2EE application platform, based on code published in Expert One-on-One J2EE Design and Development by Rod Johnson (Wrox, 2002).

Spring includes:

- ***The most complete lightweight container**, providing centralized, automated configuration and wiring of your application objects. The container is non-invasive, capable of assembling a complex system from a set of loosely-coupled components (POJOs) in a consistent and transparent fashion. The container brings agility and leverage, and improves application testability and scalability by allowing software components to be first developed and tested in isolation, then scaled up for deployment in any environment (J2SE or J2EE).*

- ***A common abstraction layer for transaction management**, allowing for pluggable transaction managers, and making it easy to demarcate transactions without dealing with low-level issues. Generic strategies for JTA and a single JDBC DataSource are included. In contrast to plain JTA or EJB CMT, Spring's transaction support is not tied to J2EE environments.*

- ***A JDBC abstraction layer** that offers a meaningful exception hierarchy (no more pulling vendor codes out of SQLException), simplifies error handling, and greatly reduces the amount of code you'll need to write. You'll never need to write another finally block to use JDBC again. The JDBC-oriented exceptions comply to Spring's generic DAO exception hierarchy.*

- ***Integration with Toplink, Hibernate, JDO, and iBATIS SQL Maps**: in terms of resource holders, DAO implementation support, and transaction strategies. First-class Hibernate support with lots of IoC convenience features, addressing many typical Hibernate integration issues. All of these comply to Spring's generic transaction and DAO exception hierarchies.*

- ***AOP functionality**, fully integrated into Spring configuration management. You can AOP-enable any object managed by Spring, adding aspects such as declarative transaction management. With Spring, you can have declarative transaction management without EJB... even without JTA, if you're using a single database in Tomcat or another web container without JTA support.*

- ***A flexible MVC web application framework**, built on core Spring functionality. This framework is highly configurable via strategy interfaces, and accommodates multiple view technologies like JSP, Velocity, Tiles, iText, and POI. Note that a Spring middle tier can easily be combined with a web tier based on any other web MVC framework, like Struts, WebWork, or Tapestry.*

The exams are one of the most important parts of this thesis, here it is how Eduard defined the exams.
Exams are defined by templates, it means that exists a predefined number of exam skeletons that allow to create all kind of exams and mapping them to the templates.
This way of work simplifies the creation of a game environment and makes much more easier to test the correctness of a map-game. It also helps for defining the logic of the game and how the user will solve the exam.

Several predefined templates:

One mandatory part
Two mandatory parts
Three mandatory parts
Choosing one of two parts
Choosing one of three parts
One mandatory part + choosing one of two parts
One mandatory part + choosing one of three parts
Choosing one of two parts + choosing one of three parts
One mandatory part + choosing one of two parts + choosing one of three parts

Table 3.1: Definition of Templates

In the current state of the application is just working the first template. We will maintain this, because it is not the focus of our thesis to create the support for the different templates.
All the exams belong to a different subject such as Maths, History, Czech, Biology. They consists of groups of partial tests that put together create an Exam.

3.3. Visual Representation of Templates

Each circle represents a collection of questions, in order to pass the exam you have to reach the final point.

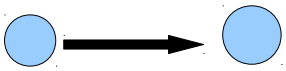
One Mandatory Part	Two Mandatory Parts	Three Mandatory Parts
		

Table 3.2: Mandatory Templates

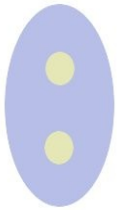
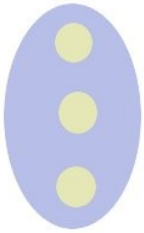
Choosing one of two Parts	Choosing two of three Parts
	 *N to M*

Table 3.3: Simple Choosing Templates

Combination of templates

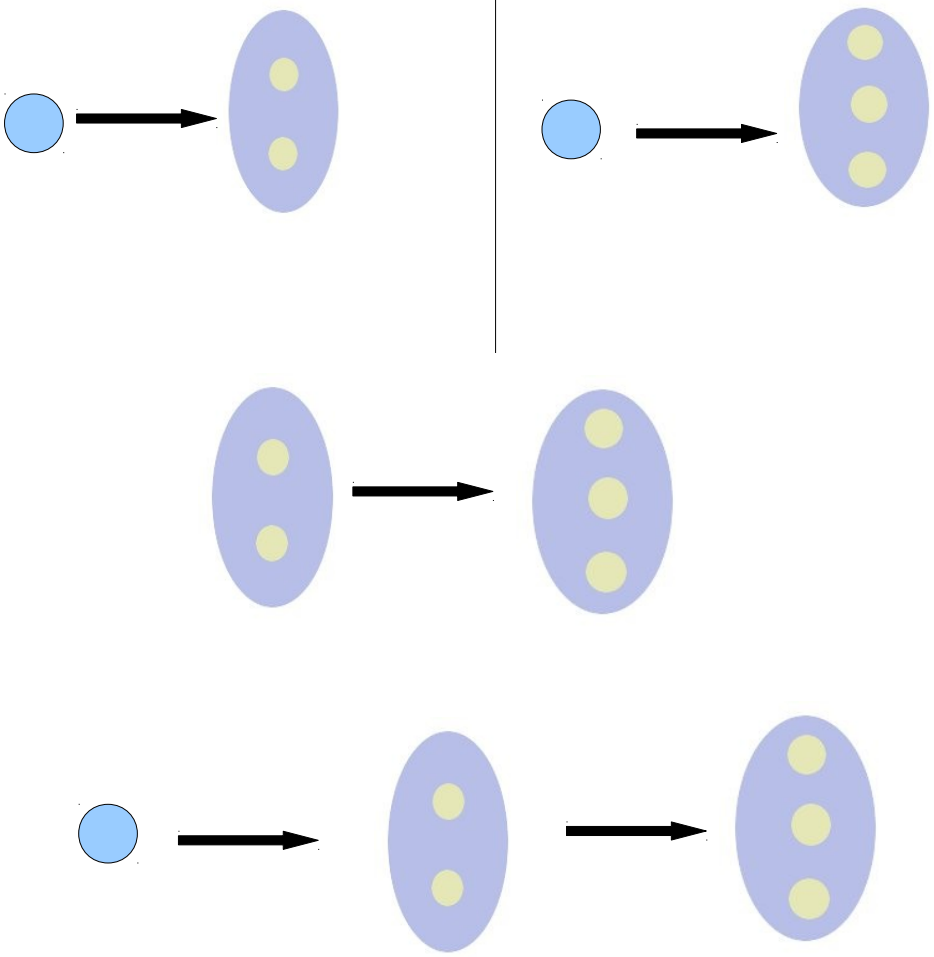


Table 3.4: Combination of Exams Templates

Here follows the schema of the database that shows how the system works and how the exams, users, questions and test are represented.

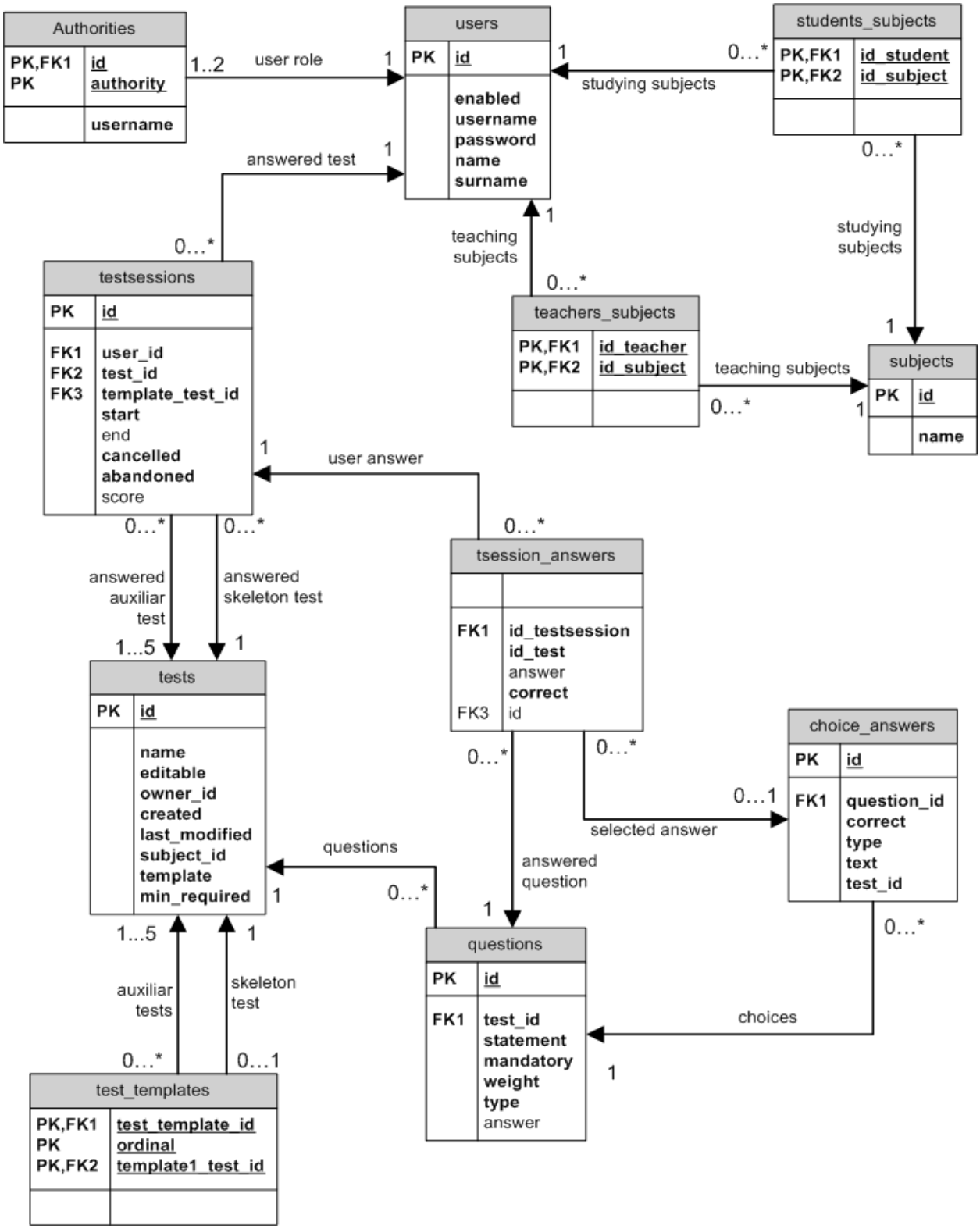


Illustration 3.3.1: Database Schema (by Eduard Martínez)

4. Application Overview

As this thesis is a part of a bigger project we are going to use some UML[xiii] diagrams for analyzing and identifying the most important areas of this thesis.

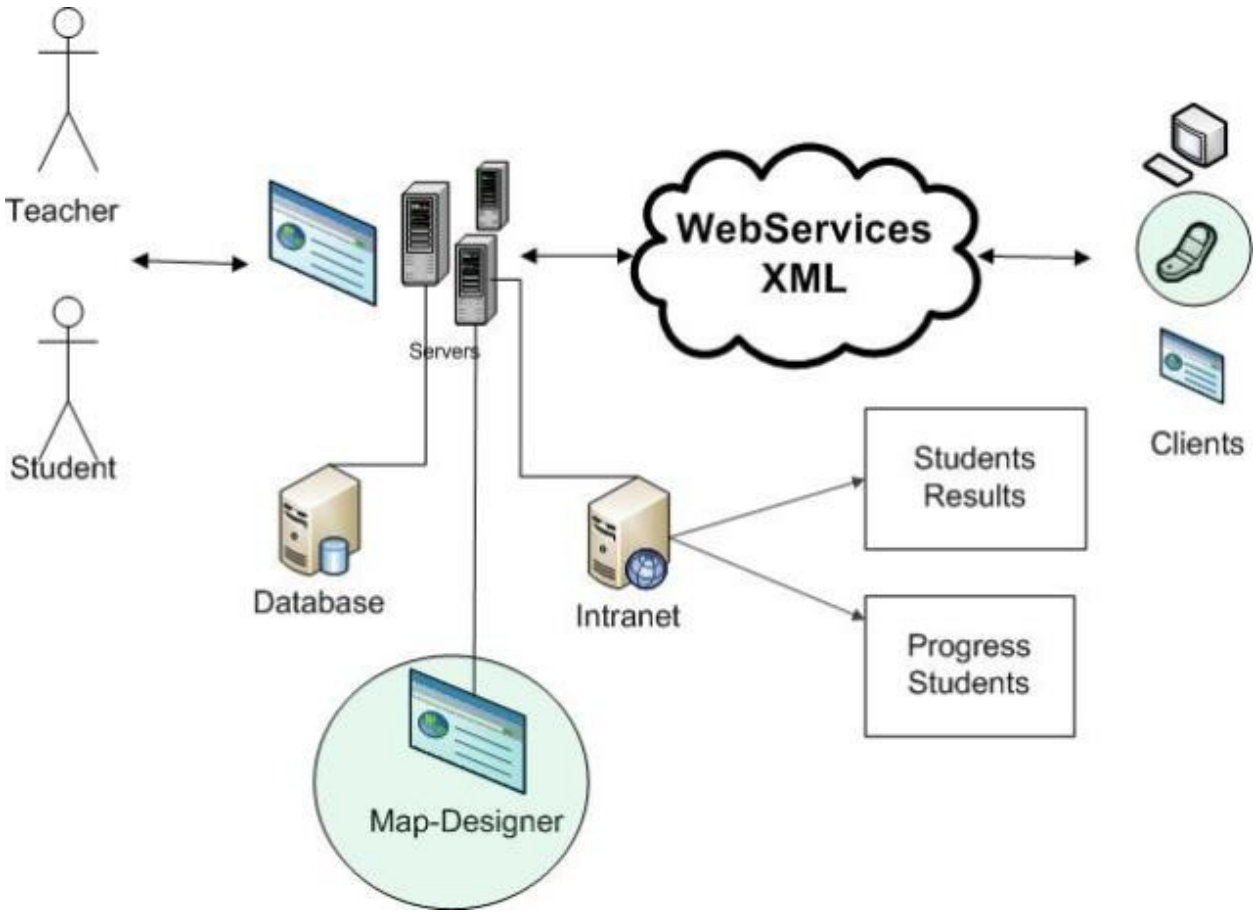


Illustration 4.1: Application Overview

In this diagram we can see the architecture of the application, the two circles that are rounding the Web Based Map Designer and the mobile client are the main focus of this thesis.
We see how the two previous theses fit in this diagram. The Antonio's thesis it is one of the clients, the computer based one. On the other hand Eduard's thesis consists of the server part of the application, managing the database and the intranet for the students.
It also takes care of the communication servers, but as we explained before we are going to change it and redo it again.

5. Communications

The communication between server and clients have to be simple,efficient,fast and cross-platform. Using XML Web-Services solve us the platform issue, now the application is defined by SOAP Web-Services is it the best option?

5.1. Communications Analysis

First of all we are going to analyze which are the critical communications and how are going to be the connections from the server to the clients.

1-Way communication	
Server to Clients	Clients to Server
Game Map Questions Game Logic	Results

Table 5.1: Table of Communications

All these communications are needed to be done at the beginning of the game execution and when the game is finished. It means that we don't need to be constantly sending information between the server and the clients.
Another point to consider is the complexity of the information. We've been discussed the low-complexity of the map, the questions are just a bunch of strings and the game logic it is represented by the template. In other words we have an amount of low-complexity information to send and receive through Web-Services.

5.2. SOAP (Simple Object Access Protocol)

Definition

SOAP, originally defined as, Simple Object Access Protocol is a protocol specification for exchanging structured information in the implementation of Web Services in computer networks. It relies on eXtensible Markup Language (XML) as its message format, and usually relies on other Application Layer protocols (most notably Remote Procedure Call (RPC) and HTTP) for message negotiation and transmission.

SOAP Structure

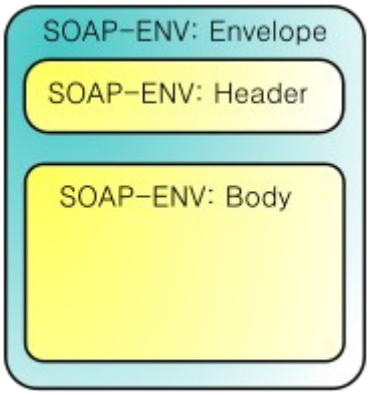


Illustration 5.2.1: Soap Structure

SOAP Characteristics
Over SMTP and HTTP protocol
XML Format
Information Overhead
Versatile
Language APIs
Complex

Table 5.2: Soap Characteristics

5.3. REST Web-Services

Rest is an architectural style developed in parallel with HTTP 1.0 protocol. It is the basis of the World Wide Web. REST exemplifies how the Web's architecture emerged by characterizing and constraining the macro-interactions of the four components of the Web, namely origin servers, gateways, proxies and clients, without imposing limitations on the individual participants. As such, REST essentially governs the proper behavior of participants.

Rest Characteristics
Client-server
Stateless
Cacheable
Layered System
Code on demand
Uniform

Table 5.3: Rest characteristics

Comparative between SOAP-RPC and Restful Web-Services	
SOAP	REST
More Complex Reinventing HTTP capabilities Extra Information New Developing Library Better tools to check integrity Flexible	Simplicity Use of HTTP capabilities SVG-Information XML libraries Difficult to check integrity

Table 5.4: SOAP vs REST

Looking at this table and thinking about what are the necessities of our application we strongly recommend to discard the current SOAP protocol and start sending the information by REST Web Services.
As the maps are going to be represented by SVG[xiv], it means a standard XML, and all the clients have the capability of reading this, it is obvious that we do not need to add any other information to the file or any other functionality from the server.
Because of that and the other reasons explained before we strongly suggest to develop the services as REST web services instead of SOAP based web services.

6. Web Services in Detail

One of the most important factors of this thesis is the definition of several Web Services that allow the server to communicate with different clients and other services.
In order to specify this REST Web Services, we will use a **WADL** description language.

6.1. WADL

6.1.1. Definition

*The **Web Application Description Language (WADL[xv])** is an XML-based file format that provides a machine-readable description of HTTP-based web applications. These applications are typically REST web services. WADL is a W3C Member Submission.*

The purpose of WADL is to allow services on the Internet (or any other IP network) to be described in a machine processable way, to make it easier to create Web 2.0 style applications and create a dynamic way of creating and configuring services. Prior to this, it was necessary to go to an existing web service, study it and write the application manually. WADL can be thought of as the REST equivalent of Web Services Description Language version 1.1. Version 2.0 of WSDL can be used to describe REST Web services, thus competing with WADL.

6.1.2. Example

In this snippet of code we can see how we define a resource for our server. First of all we have to define the URL base.
After that we can add as much resources as we want just defining a new sub-path for them. Finally we define how they work for the different four HTTP methods, such as GET,POST,DELETE and PUT.

In this case we are defining this:

- URL Base: http://tesis.com
  - Resource 1: http://tesis.com/games
    - Method: GET
    - Return: List of available games in XML format.
    - Error: 400, 404
  - Resource 1.1: http://tesis.com/games/idGame
    - Method: GET
    - Return: Information about the "idGame" in XML format
    - Error: 400, 404

```
 <resources base="http:/tesis.com/">
       <resource path="games">
           <doc>Service for Game Information</doc>
           <method name="GET">
               <response>
                   <representation mediaType="application/xml"/>
                   <fault status="400 404" />
               </response>
           </method>
           <resource path="{idGame}">
               <param name="idGame" style="template" required="true" />
               <method name="GET">
                   <response>
                       <representation mediaType="application/xml"
element="ser:Game"/>
                       <fault status="400 404" />
                   </response>
               </method>
```

Code 6.1.1: Example Of WADL

6.1.3. List of all available Web Services

System Web Service	
URL	http://tesis.com/system
Method	GET
Return	List of all available Web Services
Format	XML

All Games Web Service	
URL	http://tesis.com/games
Method	GET
Return	List of all games
Format	XML

Table 6.1: All Games Web Service

Unique Game Web Service	
URL	http://tesis.com/games/idGame
Method	GET,POST,DELETE,PUT
Return	Get,Upgrade,Deleting or creating the game with id, idGame.
Format	XML

Table 6.2: Unique Game Web Service

All Users Web Service	
URL	http://tesis.com/users
Method	GET
Return	List of all users
Format	XML

Table 6.3: All Users Web Service

Unique User Web Service	
URL	http://tesis.com/users/idUser
Method	GET,POST,DELETE,PUT
Return	Get,Upgrade,Deleting or creating the user with id, idUser.
Format	XML

Table 6.4: Unique User Web Service

All Maps Web Service	
URL	http://tesis.com/maps
Method	GET
Return	List of all maps
Format	XML

Table 6.5: All Maps Web Service

Unique Map Web Service	
URL	http://tesis.com/maps/idMap
Method	GET,POST,DELETE,PUT
Return	Get,Upgrade,Deleting or creating the map with id, idMap.
Format	XML

Table 6.6: Unique Map Web Service

All Exams Web Service	
URL	http://tesis.com/Exams
Method	GET
Return	List of all exams
Format	XML

Table 6.7: All Exams Web Service

Unique Exam Web Service	
URL	http://tesis.com/games/idExam
Method	GET,POST,DELETE,PUT
Return	Get,Upgrade,Deleting or creating the exam with id, idExam.
Format	XML

Table 6.8: Unique Exam Web Service

All Places Web Service	
URL	http://tesis.com/places
Method	GET
Return	List of all places
Format	XML

Table 6.9: All Places Web Service

Unique Place Web Service	
URL	http://tesis.com/games/idPlace
Method	GET,POST,DELETE,PUT
Return	Get,Upgrade,Deleting or creating the place with id, idPlace.
Format	XML

Table 6.10: Unique Place Web Service

7. Architecture Overview

This chapter will focus on one of the most important parts of the system that we are developing, the architecture of the system.
We will explain how the different parts of the project are interconnected and how they communicate with each other.

As we explained above, we decided to use a REST server approach to manage the communication protocols. The use of Zend Framework, will help us to implement these services in a easy way allowing future students to upgrade and improve the application without any major difficulties.

7.1. Architecture Diagram

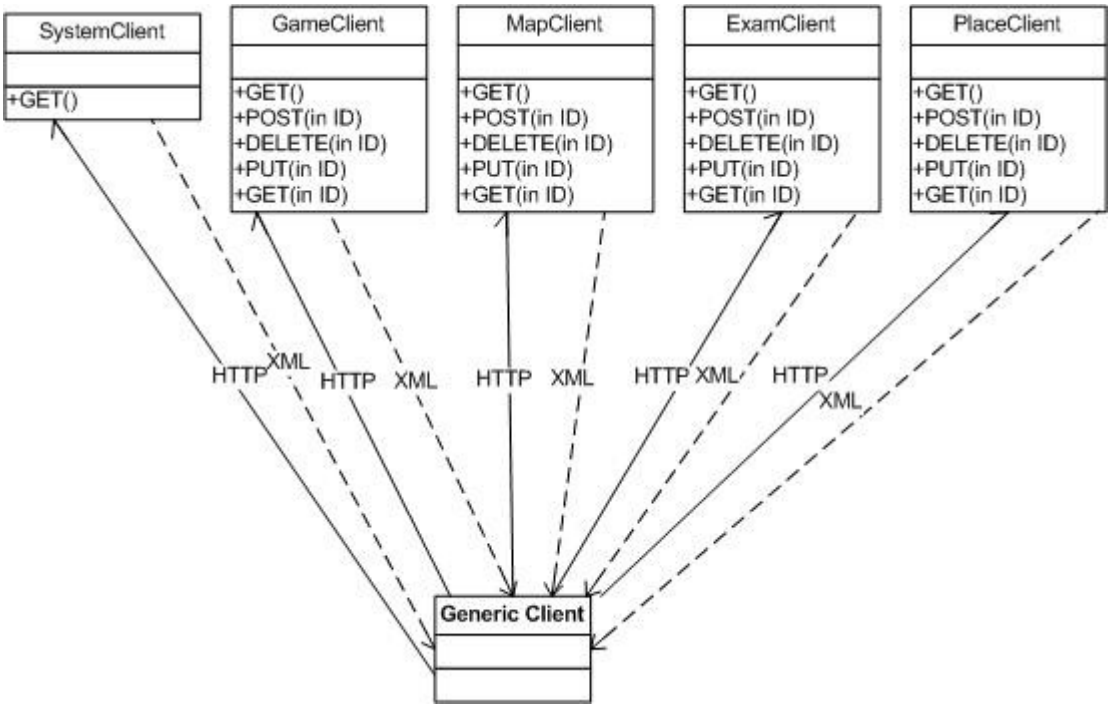


Illustration 7.1.1: Architecture of Communications

In this diagram we can see the simplicity of the architecture developed. We have a series of Controllers that manage the HTTP requests of any kind of client.

These Controllers implement the Web-Services servers of the system, and they always have the same behavior:

- Receive HTTP request
- Process request
- Generate Answer
  - Correct: Send XML data, based in one of the defined Data Structures
  - Error: 400, 404

As we can appreciate these controllers are returning a valid XML in all the different requests. We will explain how are these data defined and what they represent.

7.2. Data Structures

In this section of this work it is explained which technologies we used to define the different data that we will use. It is important that there exists a documentation of these Data models in order to facilitate the developing and testing of new clients or new services.

7.2.1. XML Schema[xvi]

Definition

*An **XML schema** is a description of a type of XML document, typically expressed in terms of constraints on the structure and content of documents of that type, above and beyond the basic syntactical constraints imposed by XML itself. These constraints are generally expressed using some combination of grammatical rules governing the order of elements, Boolean predicates that the content must satisfy, data types governing the content of elements and attributes, and more specialized rules such as uniqueness and referential integrity constraints.*

7.2.2. XML Schema Languages

For doing this task of description a XML document exists many different languages that provide us the tools to do this. The first separate one and one of the most common language is called XSD[xvii] or WXS.
We decided to choose it because it is an open W3C standard, it is really well documented, it is very flexible and it is also easy to understand as it is practically self-explicative.

Here it is the documentation of XSD taken from the wikipedia.

XSD *(XML Schema Definition)*

XML Schema, published as a W3C recommendation in May 2001, is one of several XML schema languages. It was the first separate schema language for XML to achieve Recommendation status by the W3C.

XSD can be used to express a set of rules to which an XML document must conform in order to be considered 'valid' according to that schema. However, unlike most other schema languages, XSD was also designed with the intent that determination of a document's validity would produce a collection of information adhering to specific data types. Such a post-validation infoset can be useful in the development of XML document processing software, but the schema language's dependence on specific data types has provoked criticism.

Schema and Schema Documents

Technically, a schema is an abstract collection of meta-data, consisting of a set of schema components: chiefly element and attribute declarations and complex and simple type definitions. These components are usually created by processing a collection of schema documents, which contain the source language definitions of these components. In popular usage, however, a schema document is often referred to as a schema.

Schema documents are organized by name-space: all the named schema components belong to a target name-space, and the target name-space is a property of the schema document as a whole. A schema document may include other schema documents for the same name-space, and may import schema documents for a different name-space

When an instance document is validated against a schema (a process known as assessment), the schema to be used for validation can either be supplied as a parameter to the validation engine, or it can be referenced directly from the instance document using two special attributes, `xsi:schemaLocation` *and* `xsi:noNamespaceSchemaLocation`*. (The latter mechanism requires the client invoking validation to trust the document sufficiently to know that it is being validated against the correct schema.)*

Data Types

XML Schema allows the content of an element or attribute to be validated against a data type. For example, an attribute might be constrained to hold only a valid date or a decimal number.

XSD provides a set of 19 primitive data types (boolean, string, decimal, double, float, anyURI, QName, hexBinary, base64Binary, duration, date, time, dateTime, gYear, gYearMonth, gMonth, gMonthDay, gDay, and NOTATION). It allows new data types to be constructed from these primitives by three mechanisms: restriction (reducing the set of permitted values), list (allowing a sequence of values), and union (allowing a choice of values from several types). Twenty-five derived types are defined within the

specification itself, and further derived types can be defined by users in their own schemas.

7.2.3. Data types of the system

As we can see in the definition of the Web Services of the system in previous chapters and in the illustration 8.1.1 we have a series of services that uses XML language to generate their answers.
Now we will define them with the XSD language mentioned above. We will use the graphical representation of the model to explain their structure. The exact XML schemas can be found in the appendix.

7.2.4. Exam Data Definition

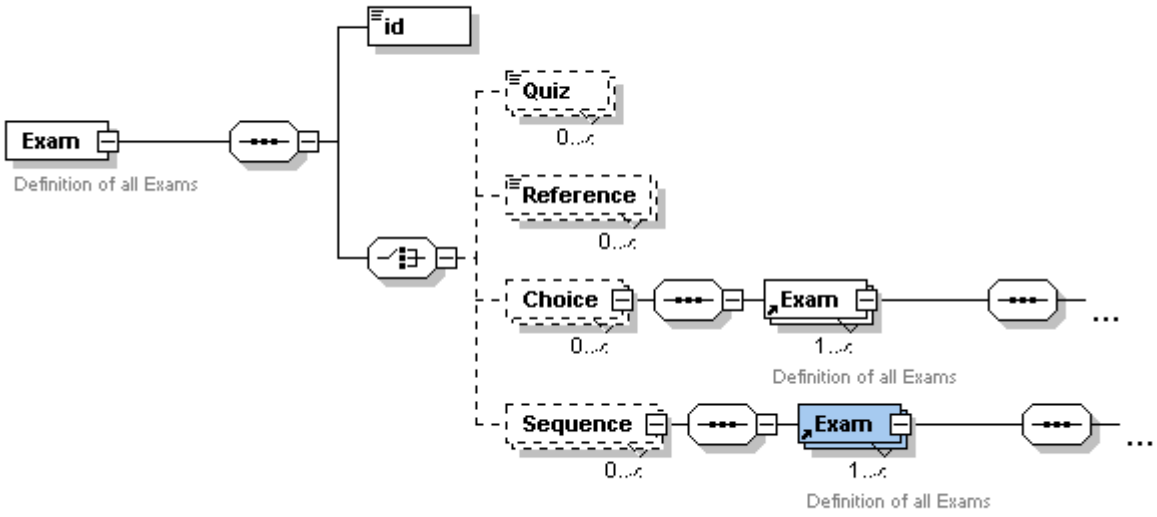


Illustration 7.2.1: Exam Model

Explanation

This schema it is quite self-explanatory so it is apparent how an Exam is defined. Every exam has a Unique ID. And can be defined in 4 different ways:

- Series of Quizzes
- Reference to any other type of structure (could be defined in the future)
- Choices of a series of Exams
- Sequence of Exams

7.2.5. Map Data Definition

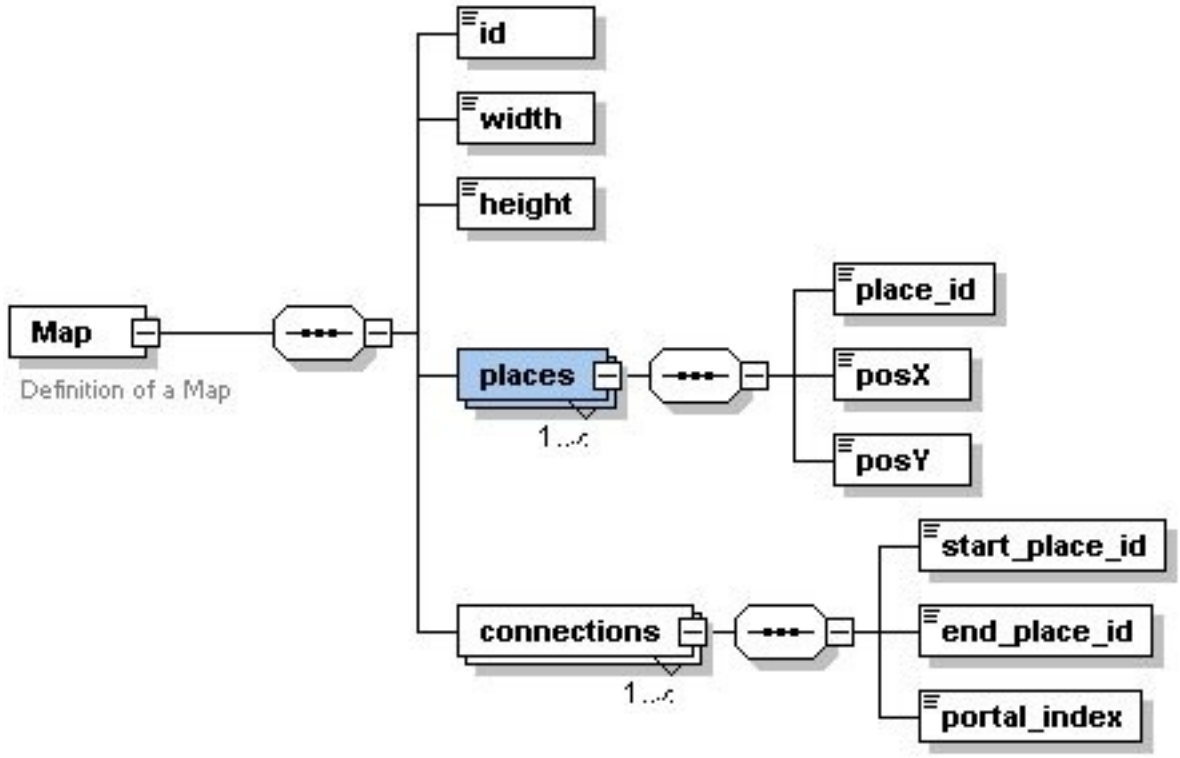


Illustration 7.2.2: Map Model

Explanation

The definition of Map we always have 3 attributes such as: id, width and height to represent the physical shape of itself.
A map basically it is a collection of places and the connections between them. As we can see above we have a sequence of places (we will discuss how they are represented bellow) and where are located in relative coordinates of the Map.
Also we have a sequence of connections (corridors) that are a subtype of places which connect the different places.

The correctness of a map doesn't depends on the definition itself. We will check the validity of the map with an algorithm at the time that we save the map.
This algorithm is explained in the Map Designer chapter.

7.2.6. Game Data Definition

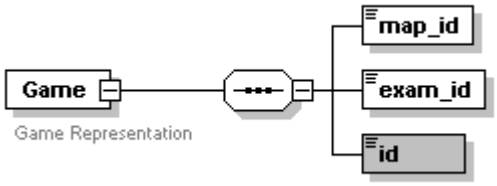


Illustration 7.2.3: Game Model

Explanation

The game representation is quite simple, a game consists of a map and an exam that fits in this map. In order to do that we need to have these two items referenced by their ids.

7.2.7. Place Data Definition

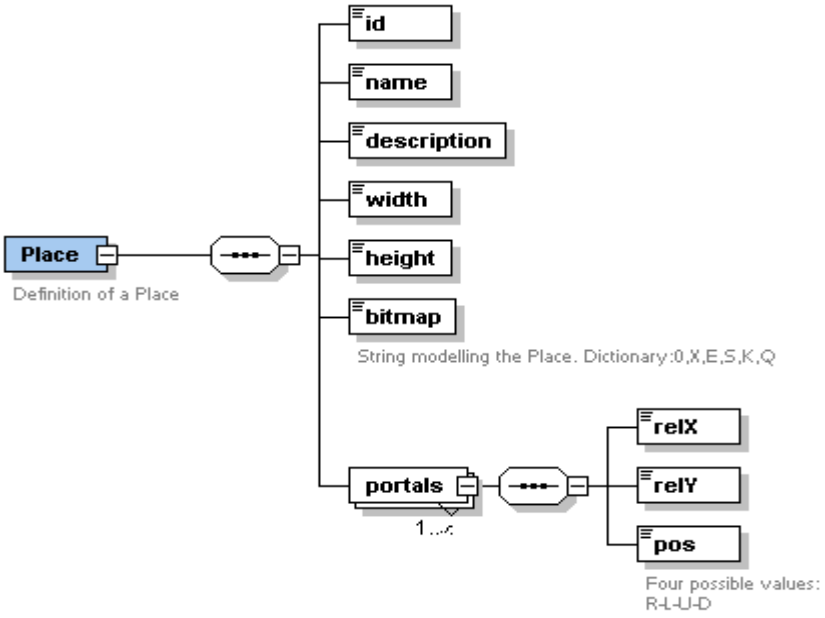


Illustration 7.2.4: Place Model

Explanation

A place is defined by the common attributes such as id,name,description, width and height.
Another attribute that is worth to mention is the bitmap. This attribute is a string of characters describing how is the room. In order to do that we created an alphabet for representing the different items that can be located in the room. We will discuss this alphabet in the Map Designer chapter.
Each place have one or more portals which are defined by relative coordinates from the room and the side of the room that the portal should be placed against.

7.2.8. System Data Definition

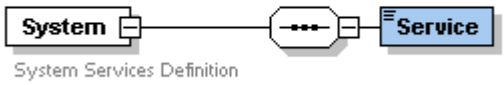


Illustration 7.2.5: System Model

Explanation

This web service is really simple. It take cares of serving the other web services addresses and configuration

7.2.9. Quiz Data Definition

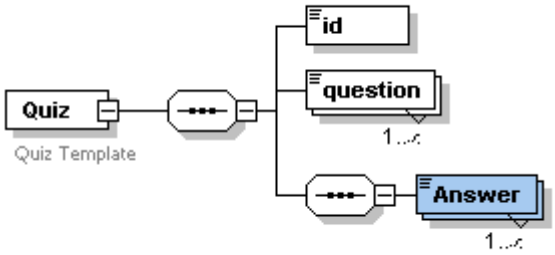
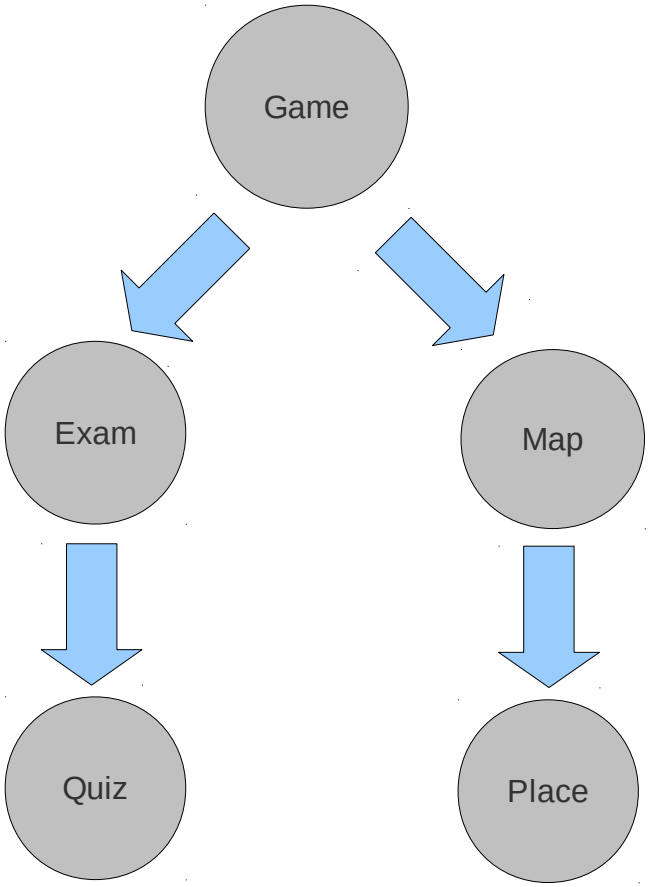


Illustration 7.2.6: Quiz Model

Explanation

A quiz is defined by an id, a question and at least one Answer. An answer can be from different types, boolean, string, integer or any other complex type previously defined.

7.3. Diagram Data Dependences



8. Implementation Overview

8.1. Zend Framework

Definition

Extending the art & spirit of PHP, Zend Framework is based on simplicity, object-oriented best practices, corporate friendly licensing, and a rigorously tested agile codebase. Zend Framework is focused on building more secure, reliable, and modern Web 2.0 applications & web services, and consuming widely available APIs from leading vendors like Google, Amazon, Yahoo!, Flickr, as well as API providers and cataloguers like StrikeIron and ProgrammableWeb.

Zend Framework it is open source licensed with the Zend Framework License[xviii] which is based in the BSD license.

Main characteristics

Simplicity, usually in the programming of a Web application we have to dedicate more than 80% of our time developing functionalities that are the same in any other application. In fact the rest 20% of the code it is what distinguishes our application of the others. Zend Framework allows us to focus on real problems letting Zend to manage all the rest.
This is how Zend Framework does it:

- Extensible and well-tested core base
- Flexible Architecture
- No configuration files necessary to get going

Web 2.0 Features

- *AJAX support through JSON – meet the ease-of-use requirements your users have come to expect*
- *Search – a native PHP edition of the industry-standard Lucene search engine*
- *Syndication – the data formats & easy access to them your Web 2.0 applications need*
- *Web Services – Zend Framework aims to be the premier place to consume & publish web services*
- *High-quality, object-oriented PHP 5 class library – attention to best practices like design patterns, unit testing, & loose coupling*

Final Choice

Even if they are more PHP frameworks completely suitable to develop this application the final decision was clearly for Zend Framework, because of its simplicity and good documentation and community behind.
Also our previous experience with this software made the choice easier so we did not have to learn another way of working and we could focus in what is real important, the development of the features of the thesis.

8.2. Web Oriented Map Designer

One of the main focus of this thesis is developing a tool for the teachers that allows them to design and create different types of maps where the exams will be represented. Taking in consideration the platform independence that we want and what is the communication protocol it is obvious that these maps have to be represented in XML as in the previous thesis.
In summary we need a technology that allows us to develop an application for designing graphically maps through the browser and save them in XML and fulfill the openness requirements.
One possible choice is Adobe Flash. With Flash technology is possible to develop a web application for designing the maps and export them to XML. It is also well documented and easy to learn. By the way it is a proprietary technology it depends on Adobe company and it is not an Open Standard.
Another choice is Scalable Vector Graphics an open standard specification for designing 2D static and animated graphics. SVG is a recommendation of W3C and it is included in the browsers since Amaya 2001, browser. Furthermore SVG is an XML based format providing us the perfect and easiest solution for developing the map designer.

8.2.1. SVG Map Designer

Looking at the whole picture, we realize that the maps designed with this tool have to represent one or various exam templates allowing the teachers to assign maps to exams.
This is a major change in the previous application logic because now it is not necessary to program some algorithms that create a Game Map from the templates. It means the application become more flexible, allow an undefined number of maps (associated to templates).

The main idea of this tool is that it should be a simple graphic designer where the teachers, not technical users, will be able to paint a 2-D map. This map have to contain all the logical information of the future game environment meaning the rooms, corridors and where the items should be placed. We will discuss all the functionalities and how they work later.

For the developing of the Map Designer we have used different approaches through the time we spent working in the Thesis.

First of all we made some research of previous web-developed applications in Internet, finding the SVG-Editor perfectly suitable for our purpose.
SVG-Editor it is a free software project, hosted in Google Code, that allows the creation of SVG graphics with a web browser. It is really complex and has a huge amount of functionalities.

Here is the definition copied directly from their website:

SVG-edit[xix] *is an on-line vector graphics editor that uses only JS, HTML5, CSS and SVG (i.e. no server-side functionality).*

SVG-edit has the following features:

Free-hand drawing Lines, Polylines Rects/Squares Ellipses/Circles Polygons/Curved Paths Stylable Text Raster Images Select/move/resize/rotate Undo/Redo Color/Gradient picker Group/ungroup Align	Zoom Layers Convert Shapes to Path Wireframe Mode Save drawing to SVG Linear Gradient Picking View and Edit SVG Source UI Localization Resizable Canvas Change Background Draggable Dialogs Resizable UI (SVG icons)	Open Local Files (Fx 3.6 only) Import SVG into Drawing (Fx 3.6 only) Connector lines and Arrows Plugin Architecture Smoother freehand paths Editing outside the canvas Increased support for SVG elements Add/edit Sub-paths Multiple path segment selection Support for foreign markup (MathML) Radial Gradients Configurable Options Eye-dropper tool Stroke linejoin and linecap Export to PNG

Table 8.1: SVG-edit Characteristics

Screenshot

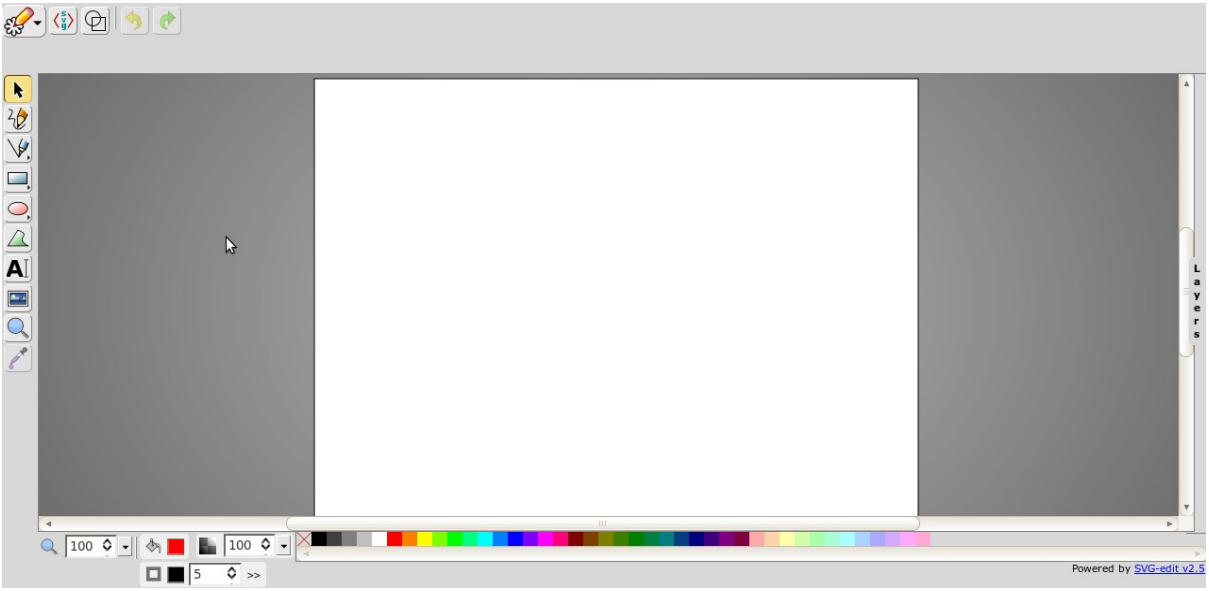


Illustration 8.2.1: Svg-Editor Screenshot

With this amount of features it is obvious that the system it is complex, it is developed in jQuery[xx] which helps a lot in the treatment of SVG and PNG graphics.
We found several problems when we were trying to adapt this software to our proposal of Map Designer

The first big issue was that this software is in constantly change and upgrading. Even if it is a good feature for a free software development in this case it was a problem because by the time we had to choose the latest stable version they were working on the new version 2.5 that will be non compatible with the previous one.

Talking about implementation issues we have to notice that the documentation of the project it is basically non existent even they have a very active and helpful Google Group.
The architecture of the version 2.4 of SVG-Edit was not module capable meaning that in order to expand the software we had to modify all the source code of the application.
After some days of research and documentation with the source code we realized that amount of time that we had to spend on it was more than the time that would take us to program the tool from the scratch.
Because of this reason we started looking possible frameworks and Javascript libraries that help the creation and managing SVG.

8.2.2. RaphaëlJS[xxi]

Raphaël is a small JavaScript library that should simplify your work with vector graphics on the web. If you want to create your own specific chart or image crop and rotate widget, for example, you can achieve it simply and easily with this library.

Raphaël [ˈrafēəl] uses the Recommendation and as a base for creating graphics. This means every graphical object you create is also a DOM object, so you can attach JavaScript event handlers or modify them later. Raphaël's goal is to provide an adapter that will make drawing vector art compatible cross-browser and easy.
Raphaël currently supports Firefox 3.0+, Safari 3.0+, Opera 9.5+ and Internet Explorer 6.0+.

8.2.3. Screen-shots and demos

8.2.4. Bitmap Manipulation

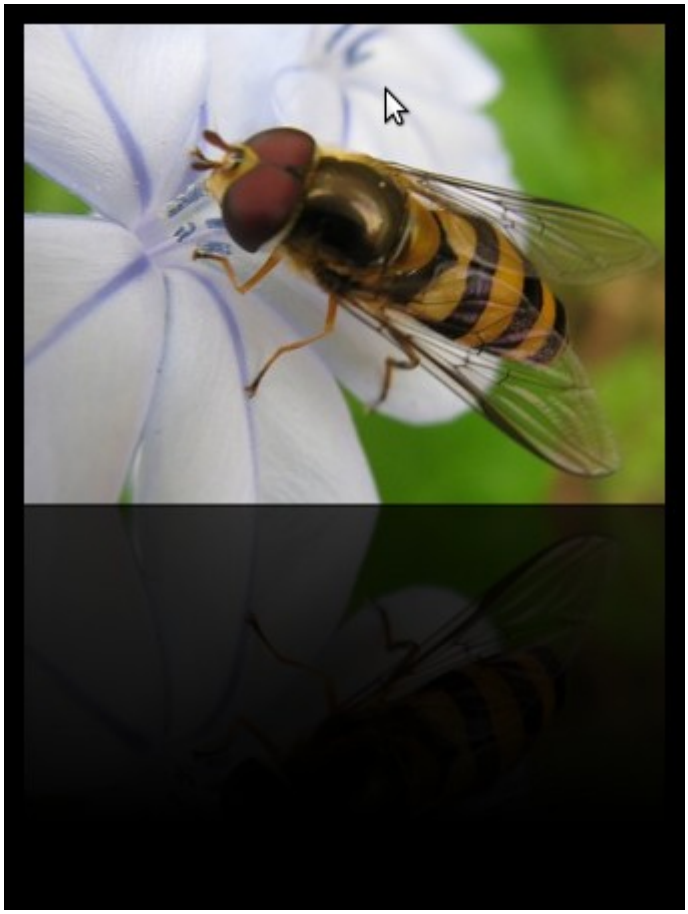


Illustration 8.2.2: RaphaelJS Reflection

8.2.5. Animation

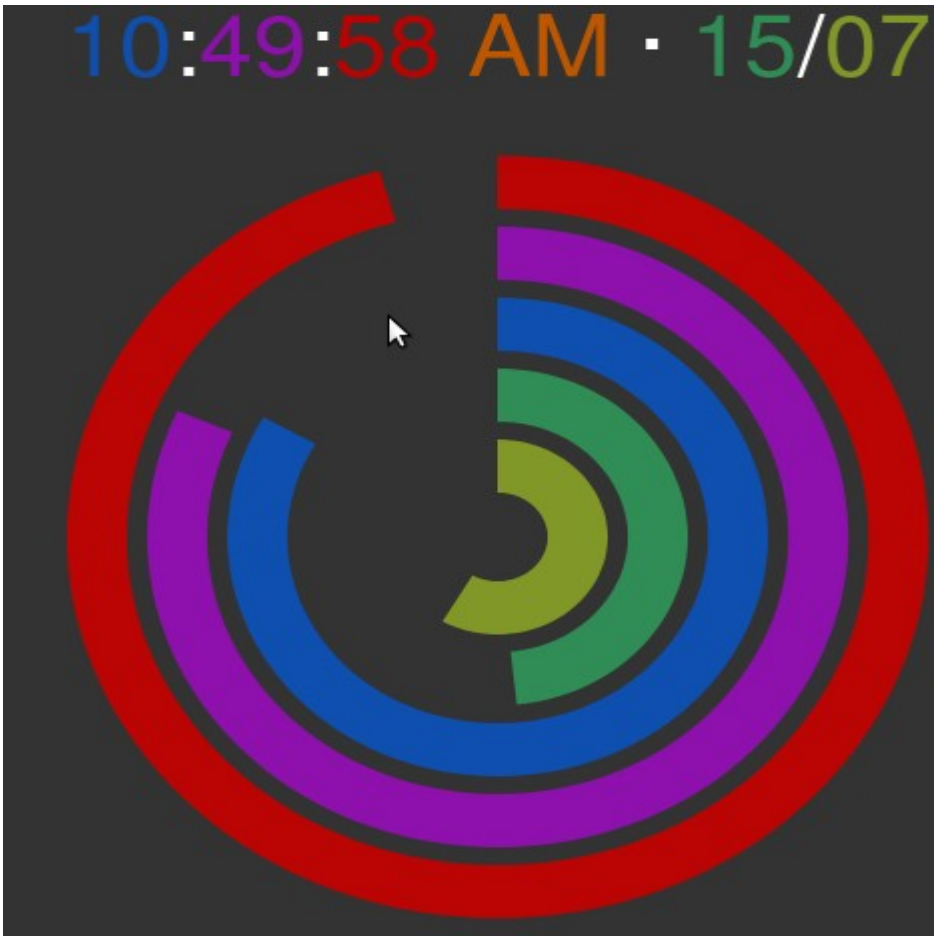


Illustration 8.2.3: RaphäelJS Animation

8.2.6. jQuery-SVG[xxii]

A jQuery plugin that lets you interact with an SVG canvas. This plugin allows you to manipulate the SVG from JavaScript.

One key factor of using jQuery-SVG is that as we are using jQuery as javascript framework for the application itself we do not have to learn another software such as Raphaël.
Even the features and characteristics are practically the same in this both systems the simplicity of jQuery and the good documentation of the plugin made us to choose this library.

8.2.7. Workflow of the Map Designer application

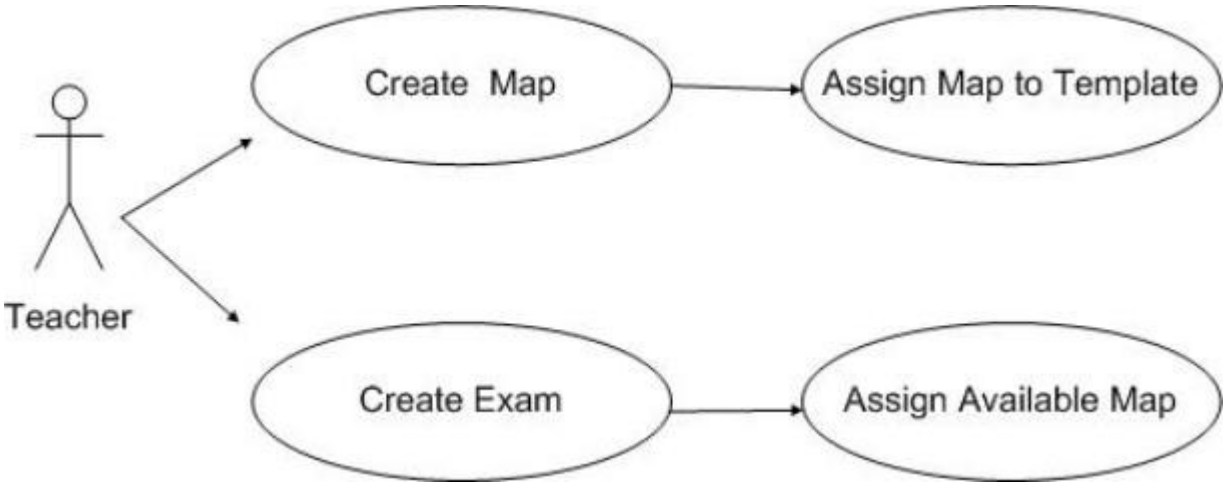


Illustration 8.2.4: Use case of Map Designer

In this simple UML diagram we can see two major differences between this approach and the previous one. Now the teachers will be able to create as many different maps as they want. For doing this there will be a database of predefined rooms and corridors the teachers just have to add to the grid the rooms taking care of it is characteristics and information.
After that, the teacher should test the correctness of the map clicking a button and link it with a suitable template of exams. This feature though it is not yet implemented, in one of the last chapter of the thesis we will do a list of functionalities that should be implemented in the future.

They will have a tool to assign these maps to possible templates. It means that the map suits for the representation of the exams that is based on this template.
When a teacher is creating an exam, the application will show which are the available maps for represent it.
This way of work save us the realization of complex heuristics algorithms in order to create dynamic maps that suits with the possible exams, also simplifying the communication between server and clients.

8.2.8. Image of Map Designer

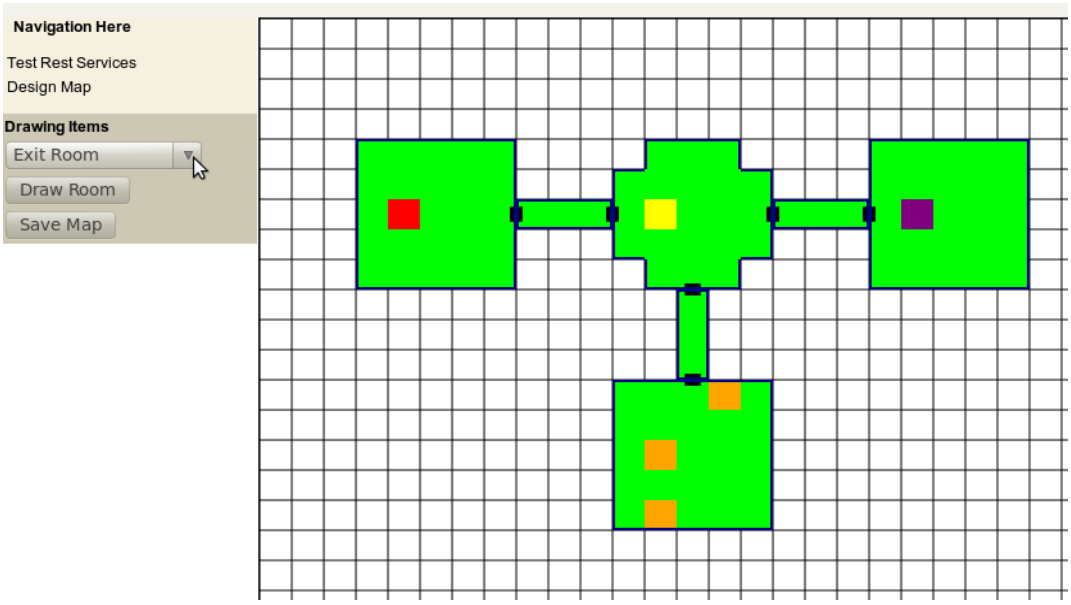


Illustration 8.2.5: Snapshot Map Designer

In this image we can take a glimpse of how this application looks like and how it works. It is really simple but it is good enough to satisfy our requirements.

At the left part, we have a list of "Drawing items" which we get from the WebServices server. The teacher just have to select one of them and it automatically will appear in the right grid.
After this, the teacher must create a valid map with several amount of these items.
As we commented before each item has some unique propierties which are defined with an XSD file. We will dicuss them in the Data Definition chapter, and there are some XML examples in the Appendix.
Finally when the teacher presses the Save Map button, the system will do the checking of correctness and save it to the server.

8.2.9. Places Implementation

As we mentioned above, in order to represent a place on the map. We defined an alphabet with a bitmap in the data structure. These are the characters recognized:

-	Empty
0	Floor
S	Starting Point
E	Exit Point
Q	Possible Question Placement
A	Possible Hint Placement

Table 8.2: Bitmap Alphabet

This alphabet is ready to be expanded in future versions.

8.2.10. Color Representation

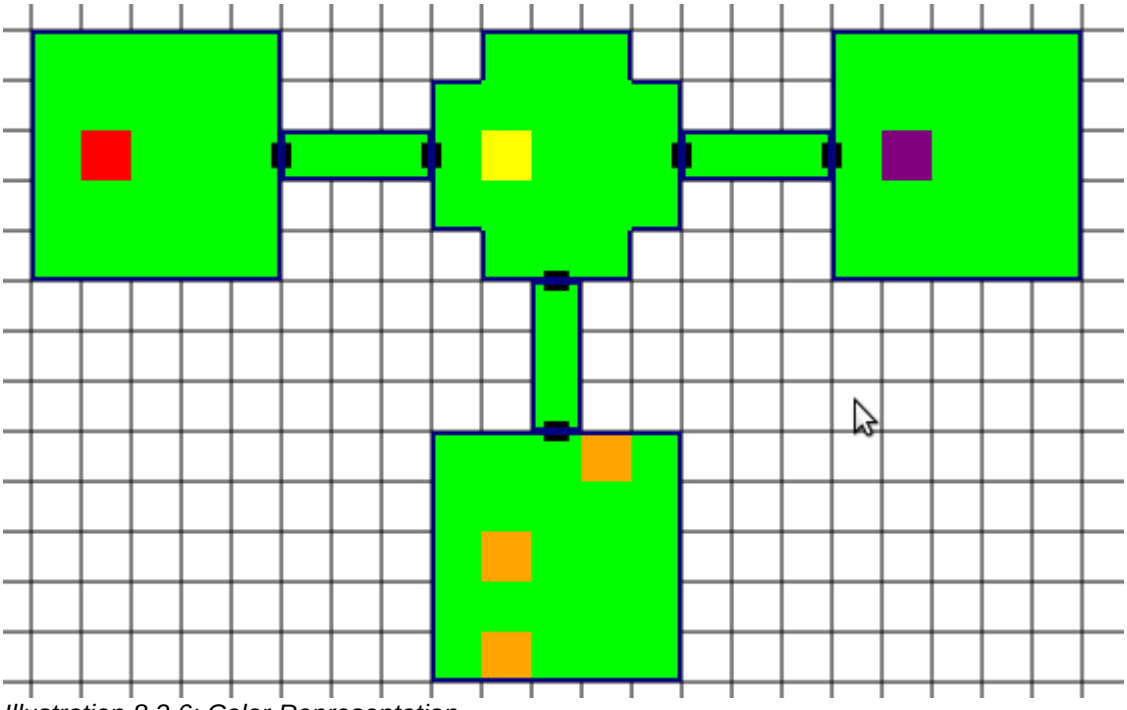


Illustration 8.2.6: Color Representation

In this map, we can appreciate seven places. Each square of a place correspond to one character of the bitmap.

Here it is the color explanation of each one:

	Floor
	Starting Point
	Hint Point
	Quiz Point
	Exit Point

Table 8.3: Color information

8.2.11. Code Snippets of Map Designer

In order to show how jQuery Ajax calls are made we demonstrate it on the two main callings to the webservices in order to get the Places defined:

Retrieving of Items

```
function getPlace(id){
     $.ajax({
              url: "http://tesis.com/places-client/"+id,
              async:false,
              success: function(xml){
                          xmlPlaces.push($(xml));
                     },
              error: function(xhr){alert(xhr.toString());},
              dataType: 'xml'
            });
}
function getPlaces(){
    for(var i = 0;i<6;i++){
        getPlace(i);
    }
}
```

Code 8.2.1: Retrieving of Items

Explanation

We can see that we have a main function “getPlaces()” that call 6 times the getPlace(id) function.
This trick is made because we have dummy data and just six predefined places, in next versions the way how the number of Places is get should be changed.

The function getPlace(id) es the one that takes care of calling the service we can appreciate that we are calling an URL, we are expecting a XML answer (dataType). And we defined the two behaviors for the answer. One in case it is working and the other in case of error.

One important thing here is to set the calling in a **synchronous** way so we are assuring that one request does not advance the older ones.

9. Mobile Client

Halfway through the development of the Thesis, we decided to include a little client for a mobile platform in order to demonstrate the correct functioning of the web services. Unfortunately the different changes and problems that we found while we were developing the Map Designer forced us to develop a dummy client.
The importance of this client,is that it can serve as an introductory tutorial for the next students who continue this Thesis so they will have a simple code that handles the REST connections and an starting code base to expand.

9.1. Mobile State of Art

The industry of mobile devices is in a great peak of innovation and market acceptation. Today there are many capable devices that can handle the system that we are developing. All of them have different Operative Systems and because of that they need an special development for each one of them.
Let us have a look of the competitors:

9.1.1. iPhone/iPod Touch - Apple

Probably the most famous devices nowadays, build by Apple Inc. they use a proprietary Operative System called IOS.
At the beginning of the Thesis the version of IOS was the 3.2, now we have the IOS 4.0 out in the market.

History

iOS[xxiii] *(previously named **iPhone OS**) is Apple's mobile operating system developed originally for the iPhone, and later deployed on the iPod Touch and iPad as well. It is derived from Mac OS X, with which it shares the Darwin foundation, and is therefore a Unix-like operating system by nature. In iOS, there are four abstraction layers: the Core OS layer, the Core Services layer, the Media layer, and the Cocoa Touch layer. The operating system uses roughly 500 megabytes of the device's storage.[1]*

Development

As iOS uses a variant of the same XNU kernel that is found in Mac OS X, the tool chain used for developing on iOS is also based on Xcode.

For developing in iOS, we need the Xcode software developing platform from Apple. This platform only runs in Mac OS in other words we need an Apple machine in order to develop an iPhone application. The main language used to program is Objective-C.

Here it is how the SDK is divided:

- Cocoa Touch:
- Media
- Core Services
- OS X Kernel

Conclusions

iPhone is a great device with a modern OS which allow us to develop a perfect client for our system. But Apple policy of privative code a close architecture and the total dependence of a company and the necessity of an Intel Mac running a Mac OS X Leopard or later in order to develop any application made us to not take iPhone in consideration.

9.1.2. Windows Mobile[xxiv] Devices

There are a lot of mobile devices that are working with the Microsoft Windows Mobile Operative System.
At first sight one of the biggest issues with these mobiles is that Microsoft it is planning to make a big change in all of them with the coming out of Windows Phone 7 series.
In other words, program now an application for the actual market phones it is programming something that is going to be obsolete in a few years.

Here is a chart with the different Windows versions

	Pocket PC 2000	Pocket PC 2002	Windows Mobile 2003	Windows Mobile 2003 SE	Windows Mobile 5.0	Windows Mobile 6	Windows Mobile 6.1	Windows Mobile 6.5
Pocket PC (Without Mobile Phone)	Pocket PC 2000	Pocket PC 2002	Windows Mobile 2003 for Pocket PC	N/A	Windows Mobile 5.0 for Pocket PC	Windows Mobile 6 Classic	Windows Mobile 6.1 Classic	N/A
Pocket PC (With Mobile Phone)	Pocket PC 2000 Phone Edition	Pocket PC 2002 Phone Edition	Windows Mobile 2003 for Pocket PC Phone Edition	Windows Mobile 2003 SE for Pocket PC Phone Edition	Windows Mobile 5.0 for Pocket PC Phone Edition	Windows Mobile 6 Professional	Windows Mobile 6.1 Professional	Windows Mobile 6.5 Professional
Smartphone (Without Touch Screen)	N/A	Smartphone 2002	Windows Mobile 2003 for Smartphone	Windows Mobile 2003 SE for Smartphone	Windows Mobile 5.0 for Smartphone	Windows Mobile 6 Standard	Windows Mobile 6.1 Standard	Windows Mobile 6.5 Standard

Illustration 9.1.1: Windows Mobile Versions (by Wikipedia)

Microsoft has been losing market share since the arrival of iPhone and Android. Even the development of an application with their Visual Studio IDE it is really a pleasure the lack of a centralized application market and the closed of their architecture made us think that Microsoft is one step behind it is competitors.

9.1.3. Android[xxv] Devices - Google

Android *is Google's Linux operating system for mobile devices. It is a competitor to the Symbian platform, Apple's iOS for the iPhone, Blackberry OS, WebOS, Maemo and Microsoft's Windows Mobile and Windows Phone for mobile devices all based on ARM architecture.*
Technologically, Android includes middleware and key applications,and uses the Linux kernel and other GNU software.It was initially developed by Android Inc., a firm later purchased by Google, and lately by the Open Handset Alliance.It allows developers to write managed code in the Java language, controlling the device via Google-developed Java libraries.
The Android operating system software stack consists of Java applications running on a Java based object oriented application framework on top of Java core libraries running on a Dalvik virtual machine featuring JIT compilation. Libraries written in C include the surface manager, OpenCore media framework, SQLite relational database management system, OpenGL ES 2.0 3D graphics API, WebKit layout engine, SGL graphics engine, SSL, and Bionic libc. The Android operating system consists of 12 million lines of code including 3 million lines of XML, 2.8 million lines of C, and 2.1 million lines of Java.
The unveiling of the Android distribution on 5 November 2007 was announced with the founding of the Open Handset Alliance, a consortium of 71 hardware, software, and telecom companies devoted to advancing open standards for mobile devices. Google released most of the Android code under the Apache License, a free software and open source license.

Because it is openness and the fact that it is not necessary any private system in order to develop an Android application, not even a mobile device to test it we would choose this alternative.
Technically speaking all the options that we explained above and even more competitors in the market that are perfectly ready to manage a client of this system such as RIM with it is blackberry, or HP-Palm with it is WebOS or some of the Symbian systems of the market can be a future gaming platform clients.

9.2. Android Client

9.2.1. Android SDK

Android its a platform that grows very fast and it is in a continuous developing. Every year there have been new versions of the SDK.
From the 1.1 that was the first really usable version until the current 2.2 it is been less than 2 years.
Here it is a table with the major versions of SDK

1.1	Released 9 February 2009
1.5 (Cupcake) Based on Linux Kernel 2.6.27	On 30 April 2009, the official 1.5 (Cupcake) update for Android was released. There were several new features and UI updates included in the 1.5 update:
1.6 (Donut) Based on Linux Kernel 2.6.29	On 15 September 2009, the 1.6 (Donut) SDK was released.
2.0/2.1 (Eclair) Based on Linux Kernel 2.6.29	On 26 October 2009 the 2.0 (Eclair) SDK was released
2.2 (Froyo) Based on Linux Kernel 2.6.32	On 20 May 2010 the 2.2 (Froyo) SDK was released.

Table 9.1: Android Versions

9.2.2. Environment Set Up

Developing for Android it is a really easy process. There are so many tutorials and official information in the Android Official website [http://developer.android.com/sdk/installing.html]

In the documentation web page that supports this document it is possible to find all the useful links for installing the SDK.

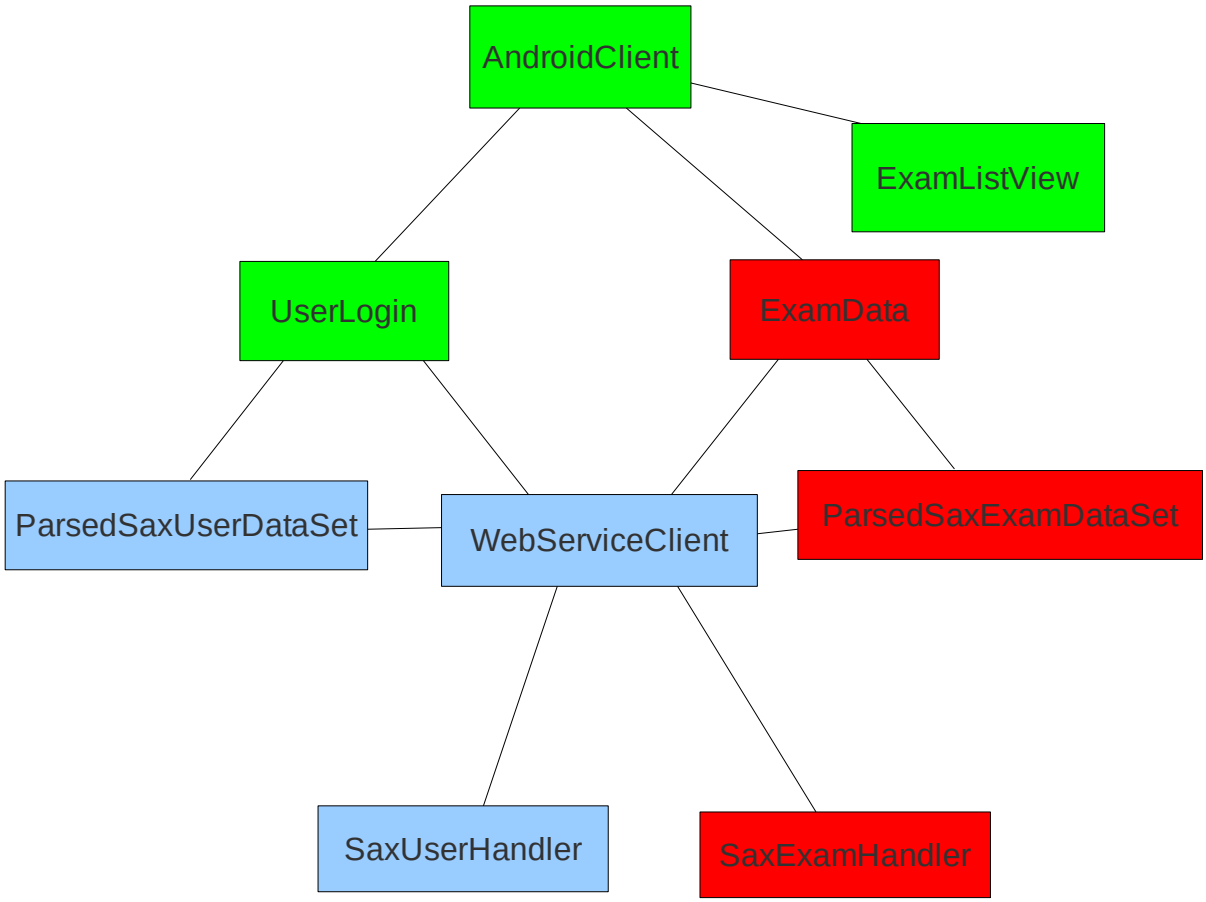
As this information changes continuously we do not consider necessary to add it in this document; in place of that we are going to explain what we really programmed and which were our biggest issues to solve during the developing of the Android Client.

9.2.3. Implementation

For developing in Android we have to start learning some basics about its architecture and the way how it is programmed.
It is not the aim of this Thesis to write a tutorial of developing in Android you can find all useful links to start programming in Android at the documentation web page.

Diagram of Classes

We will show with an informal diagram the relationship between the classes that we implemented



9.2.4. Logic of the application

These are the classes that we developed, the colors have a different meaning:

Green Classes are called *activities* and in Android SDK that means that they are launched in a different thread and they are associated to a screen.
We can see that right now we just have three different screens. We will talk later of how it is defined the style of the screen in Android.
The red Classes are in this color because they are not finished, we couldn't make it on time because we had to focus in a major change during the developing of the web services.
In fact, we can see that they are two branches of classes for the Exam part, the ExamListView class it is just a trick to show the list of Exams, but it is not calling any web service at all.

With this diagram we can appreciate how it is working the calling and the parsing of an XML file. We will use the login case because is the one that is working perfectly.

From the main activity AndroidClient we launch a new one called UserLogin. This class gets the User and Password and create a WebServiceClient in order to the call to the web service.
After that it is necessary to create a handler and a data structure for each XML that we want to read.

Visual Representation

Android it is build following a Model-View-Controller architecture, in order to create the visual aspect of the application we just have to define it in a XML file and the SDK we will take care of translating it in a suitable Java code.

Static Strings

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="title">Welcome to CastleQuest</string>
    <string name="app_name">CastleQuest CVUT</string>
<string name="data"></string>

<string name="welcome">Please Log-in</string>
    <string-array name="exams_array">
        <item>Exam 0</item>
        <item>Exam 1</item>
        <item>Exam 2</item>
        <item>Exam 3</item>
        <item>Exam 4</item>
        <item>Exam 5</item>
        <item>Exam 6</item>
    </string-array>

</resources>
```

The file strings.xml has defined all the static strings values that will appear during the execution of the application.

Layouts

All the screens that we are defining have to have a template associated to them. In this template we will define the objects such as Text Editors, Lists and any other graphical widget that we need.

Here it is the example of the template that shows the user/password login:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
   android:orientation="vertical"

  >

  <TextView android:layout_width="fill_parent"
                      android:layout_height="wrap_content"

                      android:text="Username" />

  <EditText android:id="@+id/usr"
                 android:layout_width="fill_parent"
                      android:layout_height="wrap_content"
                      />

<TextView android:layout_width="fill_parent"
                      android:layout_height="wrap_content"
                      android:text="Password"
                      />

 <EditText android:id="@+id/psw"
                 android:layout_width="fill_parent"
                      android:layout_height="wrap_content"
                      android:password="true"
                      />

  <Button android:id="@+id/btLoginDone"
               android:layout_width="wrap_content"
                      android:layout_height="wrap_content"
                      android:text="Login" />
</LinearLayout>
```

As we can see we have defined a Linear Layout for this screen, and we have added to it TextView components that act like a label, and EditText components which are the responsible of get value of the user and password. Finally we have the Login button.

Conclusions

As we can see the major problem to develop an Android application is to know well the SDK. It is very powerful and very complex so for beginners in Java language it could require more time than expected.
Once It is understood how it works the vast amount of libraries and the enormous community and documentation makes developing in Android a real pleasure.

9.2.5. Screenshots

First Screen

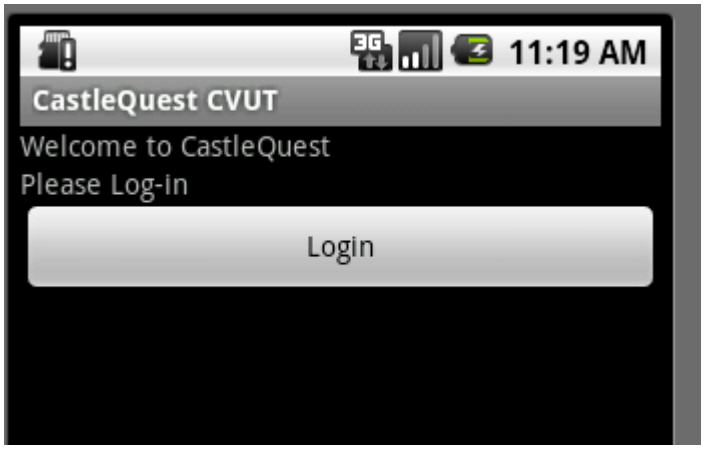


Illustration 9.2.1: Main Android Screen

Login Screen

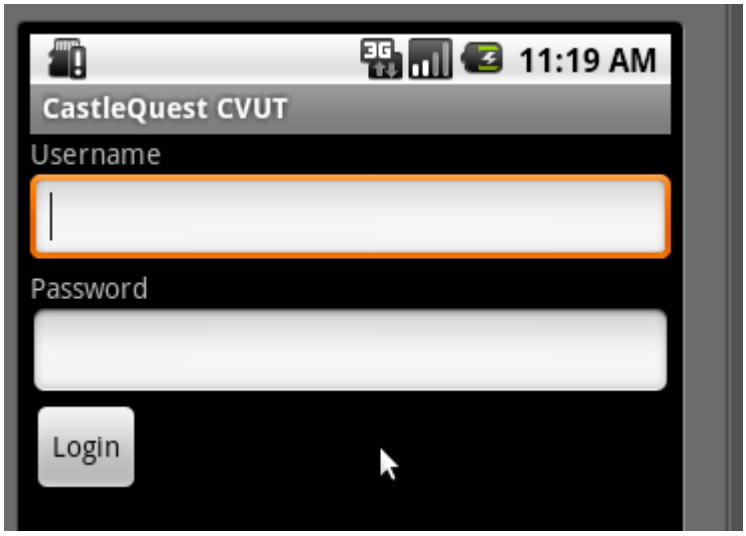


Illustration 9.2.2: User and Password

User after login

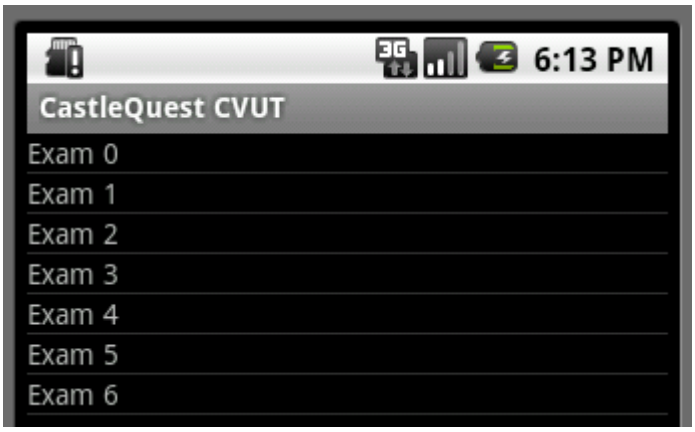


Illustration 9.2.3: List of Exams

10. Future Expansions

It is clear that the work is not complete yet. For the future successors to this thesis we will explain which are the most important points to continue with.

10.1. Database

This part of the system is developed in the Eduard Martínez thesis. It is necessary to copy the database and create the PHP classes in order to manage the database. Another option it is trying to adapt the Java code already developed by Eduard; however, we do not consider that as best idea due to bigger complexity compared to learning Java,Spring, and SOAP it is more time consuming than porting it to PHP.

10.2. Web Services

The skeleton of all the web services is created. Nevertheless as the database part is no created yet, it is necessary to upgrade the classes of the web services.
Add any other web service it is plain and simple and won't affect in any case to the already programmed.

10.3. Data Structures

The data structures defined are just a beginning, the system is ready to be expanded with different data types in order to define new maps and scenarios.

One example of this is the alphabet created to represent a Place, in more complex maps it is going to be necessary to add new characters to the alphabet.

10.4. Map Designer

The map designer need to be polished in some aspects. As the places data is currently dummy data, all the Javascript part that takes care need to be reviewed for the final version.
The major issue to face it is the algorithm that merge an Exam with a Map. For lack of time we could not implement this part and it is just developed the topological correctness checking.

11. On-line Documentation

Complementing this document and the source code we have developed a documentation web page hosting the whole project. It was done for being a place in which we can find all the information related to this thesis.

The URL address is: https://nit.felk.cvut.cz/~rafael/DOC/

12. Appendixes

12.1. XSD Files

12.1.1. Exam XSD

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSpy v2010 rel. 2 (http://www.altova.com) by Rafael (CVUT) -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
  <xs:element name="Exam">
    <xs:annotation>
        <xs:documentation>Definition of all Exams</xs:documentation>
    </xs:annotation>
        <xs:complexType>
          <xs:sequence>
            <xs:element name="id" type="xs:string"/>
                <xs:choice>
                  <xs:element name="Quiz" minOccurs="0" maxOccurs="unbounded"/>
                  <xs:element name="Reference" minOccurs="0" maxOccurs="unbounded"/>
                  <xs:element name="Choice" minOccurs="0" maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:sequence>
                          <xs:element ref="Exam" maxOccurs="unbounded"/>
                        </xs:sequence>
                    </xs:complexType>
            </xs:element>
                  <xs:element name="Sequence" minOccurs="0" maxOccurs="unbounded">
                    <xs:complexType>
                      <xs:sequence>
                          <xs:element ref="Exam" maxOccurs="unbounded"/>
                      </xs:sequence>
                    </xs:complexType>
                  </xs:element>
                </xs:choice>
          </xs:sequence>
        </xs:complexType>
  </xs:element>
</xs:schema>
```

12.1.2. Game XSD

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSpy v2010 (http://www.altova.com) by Rafael (CVUT) -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
	<xs:element name="Game">
		<xs:annotation>
			<xs:documentation>Game Representation</xs:documentation>
		</xs:annotation>
		<xs:complexType>
			<xs:sequence>
				<xs:element name="map_id" type="xs:string"/>
				<xs:element name="exam_id" type="xs:string"/>
				<xs:element name="id"/>
			</xs:sequence>
		</xs:complexType>
	</xs:element>
</xs:schema>
```

12.1.3. Place XSD

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSpy v2010 rel. 2 (http://www.altova.com) by Rafael (CVUT) -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
	<xs:element name="Place">
		<xs:annotation>
			<xs:documentation>Definition of a Place</xs:documentation>
		</xs:annotation>
		<xs:complexType>
			<xs:sequence>
				<xs:element name="id" type="xs:string"/>
				<xs:element name="name" type="xs:string"/>
				<xs:element name="description" type="xs:string"/>
				<xs:element name="width" type="xs:integer"/>
				<xs:element name="height" type="xs:integer"/>
				<xs:element name="bitmap" type="xs:string">
					<xs:annotation>
			<xs:documentation>String modelling the Place. Dictionary:0,X,E,S,K,Q
					</xs:documentation>
					</xs:annotation>
				</xs:element>
				<xs:element name="portals" maxOccurs="unbounded">
					<xs:complexType>
						<xs:sequence>
							<xs:element name="relX" type="xs:integer"/>
							<xs:element name="relY" type="xs:integer"/>
							<xs:element name="pos" type="xs:string">
								<xs:annotation>
				<xs:documentation>Four possible values: R-L-U-D</xs:documentation>
								</xs:annotation>
							</xs:element>
						</xs:sequence>
					</xs:complexType>
				</xs:element>
			</xs:sequence>
		</xs:complexType>
	</xs:element>
</xs:schema>
```

12.1.4. Map XSD

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSpy v2010 rel. 2 (http://www.altova.com) by Rafael (CVUT) -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
	<xs:element name="Map">
		<xs:annotation>
			<xs:documentation>Definition of a Map</xs:documentation>
		</xs:annotation>
		<xs:complexType>
			<xs:sequence>
				<xs:element name="id"/>
				<xs:element name="width"/>
				<xs:element name="height"/>
				<xs:element name="places" maxOccurs="unbounded">
					<xs:complexType>
						<xs:sequence>
							<xs:element name="place_id"/>
							<xs:element name="posX"/>
							<xs:element name="posY"/>
						</xs:sequence>
					</xs:complexType>
				</xs:element>
				<xs:element name="connections" maxOccurs="unbounded">
					<xs:complexType>
						<xs:sequence>
							<xs:element name="start_place_id"/>
							<xs:element name="end_place_id"/>
							<xs:element name="portal_index"/>
						</xs:sequence>
					</xs:complexType>
				</xs:element>
			</xs:sequence>
		</xs:complexType>
	</xs:element>
</xs:schema>
```

12.1.5. Quiz XSD

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSpy v2010 (http://www.altova.com) by Rafael (CVUT) -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
	<xs:element name="Quiz">
		<xs:annotation>
			<xs:documentation>Quiz Template</xs:documentation>
		</xs:annotation>
		<xs:complexType>
			<xs:sequence>
			  <xs:element name="id" type="xs:string"/>
			  <xs:element name="question" type="xs:string" maxOccurs="unbounded"/>
			     <xs:sequence>
				<xs:element name="Answer" type="xs:anySimpleType" maxOccurs="unbounded"/>
				     </xs:sequence>
			</xs:sequence>
		</xs:complexType>
	</xs:element>
</xs:schema>
```

12.1.6. System XSD

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSpy v2010 (http://www.altova.com) by Rafael (CVUT) -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
	<xs:element name="System">
		<xs:annotation>
			<xs:documentation>System Services Definition</xs:documentation>
		</xs:annotation>
		<xs:complexType>
			<xs:sequence>
				<xs:element name="Service" type="xs:string"/>
			</xs:sequence>
		</xs:complexType>
	</xs:element>
</xs:schema>
```

i PHP – PHP: Hypertext processor – a server-side scripting language Http://www.php.net
ii Javascript- Prototype Scripting Language http://en.wikipedia.org/wiki/JavaScript
iii Cascading Style Sheets (CSS) http://www.w3.org/Style/CSS/
iv Simple Object Access Protocol http://en.wikipedia.org/wiki/SOAP
v Java http://www.java.com/
vi Zend Framework http://framework.zend.com/
vii C# http://en.wikipedia.org/wiki/C_Sharp
viii Direct 3D http://en.wikipedia.org/wiki/Direct3D
ix OpenGL http://en.wikipedia.org/wiki/Opengl
x Spring Framework http://www.springsource.org/
xi XML Http://www.xml.org
xii REST http://en.wikipedia.org/wiki/Representational_State_Transfer
xiii UML http://en.wikipedia.org/wiki/Unified_Modeling_Language
xiv Scalable Vector Graphics (SVG) http://en.wikipedia.org/wiki/Svg
xv WADL http://en.wikipedia.org/wiki/Web_Application_Description_Language
xvi XML Schema http://en.wikipedia.org/wiki/XML_Schema
xvii XSD http://en.wikipedia.org/wiki/Xsd
xviii Zend License http://framework.zend.com/license
xix SVG Edit http://svg-edit.googlecode.com/
xx jQuery Framework Http://www.jquery.com
xxi Raphaël Framework http://raphaeljs.com/
xxii jQuery-SVG http://keith-wood.name/svg.html
xxiii iOS http://en.wikipedia.org/wiki/IOS_%28Apple%29
xxiv Windows Mobile http://en.wikipedia.org/wiki/Windows_mobile
xxv Android http://en.wikipedia.org/wiki/Android_%28operating_system%29